Opisywaliśmy już w INFORMATYCE system operacyjny CP/M [2]. Obecnie możemy stwierdzić, że jest to system, który najbardziej rozpowszechnił się w kraju. Dlatego też nie powinno być dużych kłopotów ze zdobyciem zawierającej go dyskietki. To jednak dopiero początek. Co zrobić dalej napisano w poniższym tekście.

Dla Czytelników mamy jeszcze niespodziankę. Ponieważ instalacja wymaga samodzielnego napisania pewnych programów, warto posłużyć się wzorcem — na przykład oprogramowaniem dla systemu MDS. Aby otrzymać kserokopię wersji źródłowej należy przesłać pod adresem redakcji:

— dużą zaadresowaną kopertę ze znaczkiem
— odcięty róg okładki z napisem mikroKLAN 7
— krótki opis posiadanego sprzętu i dotychczasowych poczynań (tylko do wiadomości redakcji).

Powyższa oferta dotyczy tylko hobbystów — osób prywatnych; ważna jest przez dwa miesiące od ukazania się numeru. Zainteresowane firmy i instytucje prosimy o kontakt bezpośredni.

Jak zainstalować CP/M

Poniższy opis dotyczy wersji CP/M 1.4. Ma on inną wielkość modułu **BDOS** i **BIOS** niż opisany w [2]. Moduł BDOS ma wielkość $D00_H$, a moduł BIOS wielkość 200_H. Dzięki temu system ten można zainstalować w systemach wyposażonych w pamięć RAM o wielkości 16 KB (a nie co najmniej 20 KB), co w przypadku konstrukcji amatorskich może być istotne. W wersji CP/M 1.4 adresy modułów dla pamięci 16 KB są następujące:

CCP — 2900_H
BDOS — 3100_H
BIOS — $3E00_H$

Adresy dla innych konfiguracji pamięci można obliczyć analogicznie jak to podano w [2]. Poniższy opis może być z powodzeniem stosowany dla innych wersji systemu CP/M. W tym przypadku, tam gdzie zostało to zaznaczone w opisie, należy korzystać z adresów podanych w [2].

Co należy mieć

1. Mikrokomputer wyposażony w:
— mikroprocesor 8080 lub odpowiednik (może być też Z80)
— terminal alfanumeryczny (np. telewizor z klawiaturą) umożliwiający wprowadzanie i wyprowadzanie znaków
— jednostkę dysków elastycznych
— pamięć RAM o pojemności min. 16 KB, umieszczoną w postaci spójnego obszaru od adresu 0.

2. Dysk elastyczny zawierający system operacyjny CP/M.

3. Środki programowe lub sprzętowe umożliwiające zapis pamięci RAM i badanie jej zawartości. Ponadto, bardzo przydatne są środki umożliwiające przygotowanie krótkich programów — do ok. 512 bajtów (np. EDYTOR i ASSEMBLER) oraz ułatwiające ich uruchamianie (np. praca krokowa i punkt wstrzymania).

4. Jeden lub dwa sformatowane, ale nie zapisane dyski elastyczne.

A oto kolejne kroki:

1. Jeżeli brak oprogramowania sterującego jednostką dysków, należy je napisać. Powinno ono umożliwić zapis i odczyt dowolnie wybranego sektora z dowolnej ścieżki dysku. Podprogram ten powinien również sygnalizować sytuacje błędne (błąd zapisu, odczytu, adresu, typu operacji itp.). Najlepiej jest umieścić go w pamięci stałej poza obszarem działania systemu CP/M.

2. Należy przygotować program odczytujący dwie pierwsze ścieżki (o adresie 0 i 1) z dysku i zapisujący je do pamięci od adresu 2880_H. Działanie tego programu należy przetestować, odczytując dwie pierwsze ścieżki z dysku o znanej zawartości — do wyzerowanej pamięci.

3. Należy przygotować program zapisujący zawartość pamięci od adresu 2880_H na dwie pierwsze ścieżki dysku i przetestować go, wykorzystując program z p. 2. Oba programy powinny być umieszczone w różnych miejscach pamięci poniżej adresu 2880_H (w przypadku nowszych wersji systemu należy zamiast adresu 2880_H w obu przypadkach użyć adresu 3380_H).

4. Jeżeli mikrokomputer nie ma programu ładowania systemu operacyjnego (np. ISIS II), to należy go przygotować i umieścić w pamięci stałej. Program ten powinien ładować do pamięci sektor nr 1 ze ścieżki nr 0 i przekazywać sterowanie do rozkazu znajdującego się w pierwszym bajcie załadowanego sektora. Jeżeli — na przykład — zamierzamy w przyszłości wykorzystywać system ISIS II lub podobny, to musimy ładować ścieżkę nr 0 od adresu 3000_H i przekazywać sterowanie pod ten adres.

5. Trzeba przygotować program ładujący system CP/M, podobny do programu z p. 2. Pewna trudność powstaje, jeżeli program z poprzedniego punktu ładuje pierwszy sektor pod adres 3000_H. Wtedy podczas ładowanie

Nie będzie polskiego komputera osobistego

Na fenomen zwany komputerem osobistym — wbrew temu, co usiłują wmówić laikom technokraci — nie składają się jedynie cudowne możliwości zaklęte w niewielkim pudełku. Pierwszorzędną rolę grają trzy zupełnie nietechniczne czynniki. Pierwszy to świadomość masowa, kształcona na powieściach science-fiction i publicystyce popołudniówek; mit elektronicznego boga zdolnego rozwiązać wszystkie problemy doczesnego życia. Drugi czynnik to łatwość dostępu: „bóstwo" można nabyć w każdym domu towarowym. Trzeci, zupełnie przyziemny aspekt to cena, zbliżona do kosztu kanapy czy naprawy samochodu.

W sferze ignorancji (pierwszy czynnik) chyba nawet przebijamy kraje zachodnie. Brak szeroko dostępnych źródeł rzetelnej, fachowej i — na dodatek — przystępnie podanej informacji sprzyja mitomanii; ba, nawet stymulowaniu snobizmu. Co gorsze — jest też jednak źródłem obaw o osobisty prestiż, o utratę prawa podejmowania „nieomylnych" decyzji. Rodzima ignorancja prowadzi do stawiania komputera w roli konkurenta, a nie narzędzia.

Rozpatrując drugi czynnik, musimy spojrzeć realnie na dotychczasową produkcję przemysłu informatycznego, który nie jest w stanie sprostać nawet zamówieniom profesjonalnych odbiorców. Masowo produkowany i sprzedawany mikrokomputer to pozbawiona najmniejszych szans mrzonka. Nie pomogą tu żadne genialne konstrukcje, programy rządowe czy nawet zakupy licencji. Przemysł informatyczny nie jest samodzielną wyspą — zależy od reszty gospodarki, o której stanie świadczy zaopatrzenie w inne „masowe" produkty.

Trzeci czynnik, czyli cena — to niestety gwóźdź do trumny. Nawet najwięksi pesymiści nie oczekują takiego wzrostu cen mebli czy usług, by sprostały one krzykliwie anonsowanym w gazetach polskim mikrokomputerom (MERITUM, ComPAN). Użycie nazwy „komputer osobisty" w stosunku do tych jednostkowo produkowanych urządzeń zakrawa na kpinę.

Warto jednak pochwalić pomysł skierowania całej produkcji MERITUM do szkół, a nie do sklepów. Zapewne musiała w tym maczać palce jakaś Federacja Ochrony Konsumenta, zabraniająca traktowania swych podopiecznych jak idiotów. Wiadomo bowiem powszechnie, że za równowartość ceny MERITUM (uwzględniając kurs czarnorynkowy) można na Zachodzie nabyć mikrokomputer o wielokrotnie lepszych parametrach.

AJP

wania program ten zostanie zniszczony. W tym przypadku musi się on najpierw przepisać pod adres niższy od 2880_H, przekazać tam sterowanie i dopiero wtedy załadować system CP/M. Program ten musi zmieścić się w 128 bajtach, ponieważ na dysku systemowym przewidziany jest dla niego tylko jeden sektor. Zawartość dysku ze ścieżki nr 0 od sektora nr 2 do końca należy załadować do obszaru pamięci od adresu 2900_H, a następnie zawartość dysku od początku ścieżki nr 1 do sektora nr 21 do obszaru pamięci od adresu 3580_H (w przypadku nowszych wersji systemu wymagających pamięci 20 KB należy załadować całe dwie pierwsze ścieżki bez pierwszego sektora pod adres 3400_H).

6. Należy przygotować moduł **BIOS**. Powinien on być skonstruowany tak jak opisano w [2], z tym że w wersji CP/M 1.4 nie są potrzebne podprogramy **LISTST** i **SECTRAN.** Jeżeli istnieją kłopoty z przygotowywaniem programów (brak programów EDYTOR i ASSEMBLER), to można także opuścić podprogramy **LIST, PUNCH** i **READER,** zwłaszcza jeżeli w systemie nie jest używana drukarka, czytnik i perforator lub inne zastępujące je urządzenia. Należy zostawić wtedy rozkaz skoku do tych podprogramów, a ich treść zastąpić rozkazem RET. Często się także zdarza, że w pamięci stałej posiadanego mikrokomputera zawarte są już podprogramy współpracy z konsolą i innymi urządzeniami peryferyjnymi. Warto je wykorzystać dopisując jedynie fragment przygotowujący parametry i odbierający wyniki, jeżeli ich specyfikacja nie jest zgodna z opisem w [2]. Do realizacji pozostałych podprogramów modułu **BIOS** można wykorzystać napisany wcześniej program współpracy z jednostką dysków (patrz p. 1).

7. Po sprawdzeniu programu z p. 5 i modułu **BIOS,** można przystąpić do utworzenia dysku z systemem CP/M. W tym celu należy:

— używając programu z p. 2 załadować do pamięci system CP/M z posiadanego dysku

— zamienić zawartość pamięci od adresu 2880_H do adresu $28FF_H$ na program przygotowany w p. 5 oraz od adresu $3E00_H$ do adresu $3FFF_H$ na nowy kod modułu BIOS przygotowany w p. 6

— zapisać na czysty dysk tak przygotowany system za pomocą programu z p. 3 (dla nowszych wersji systemu adresy 3880_H—$38FF_H$ i $4A00_H$—$4F7F_H$).

8. Przygotowany w poprzednim kroku system należy sprawdzić w sposób następujący:

— za pomocą programu z p. 4 zainicjować system. Jeżeli wszystkie poprzednie kroki były wykonane prawidłowo, to CP/M zgłasza się wyświetlając na ekranie **A>**; jeżeli system nie zgłosi się, należy jeszcze raz zbadać poprawność poprzednich kroków

— testować działanie systemu pisząc: **SAVE 1 X.COM;** po wykonaniu tej operacji system zgłasza się przez wyświetlenie na ekranie **A>**; Należy teraz napisać: **DIR;** w odpowiedzi system powinien wyświetlić na ekranie:

```
A: X .COM
A>
```

należy wtedy napisać dyrektywę kasowania: **ERA X.COM,** po wykonaniu której można ponownie dyrektywą **DIR** sprawdzić poprawność jej wykonania

— jeżeli test wypadł pomyślnie, to można skompletować dysk systemowy; wykorzystując programy z p. 2 i p. 3, należy przepisać przygotowany system na dysk zawierający jego pierwotną wersję i — wykorzystując wszystkie możliwości systemu — uzupełnić moduł **BIOS,** testując w ten sposób działanie całego systemu.

9. Nową wersję modułu **BIOS** i ewentualnie programu ładującego lub wersje systemu dla innej pamięci RAM należy przygotować tak, jak tó zostało opisane w [2]. Należy przy tym pamiętać, że dla wersji CP/M 1.4 system utworzony programem **MOVCPM** i znajdujący się również w obszarze pamięci od adresu 900_H ma program ładujący w tym samym miejscu co nowsze wersje systemu, ale moduł **BIOS** znajduje się pod adresem $1E80_H$, natomiast w docelowym systemie adresem modułu będzie adres $3E00_H+b$.

IRENEUSZ MYZIK
PIE

LITERATURA

[1] Hogan T.: Osborne CP/M User Guide. Osborne/McGraw-Hill, Berkeley, California
[2] Sobczyk M.: Dyskowy system operacyjny CP/M. INFORMATYKA, nr 2, 3, i 4, 1983.

Mimo mody (a może konieczności?) na ulepszanie i rozbudowywanie mikrokomputerów, nie radzimy nikomu przerabiać ZX81 na Mercedesa. Zamieszczony niżej program przyczynia się jednak do powiększenia komfortu pracy z tym mikrokomputerem. W „doroślejszym" ZX SPECTRUM opisane funkcje realizuje oprogramowanie producenta. ZX81 trzeba udoskonalić — by po wielogodzinnej pracy móc spokojnie zasnąć, ze świadomością, że nasz program utrwalony na kasecie nie okaże się nazajutrz tylko iluzją.

Możliwość przejrzenia kasety jest cenna nie tylko dla bałaganiarzy. Pozwala również sprawdzić, czy nasze „biblioteczne" programy są nadal w porządku. Szkoda tylko, że nikt — jak dotąd — nie wymyślił praktyki mniej czasochłonnej.

Program VTL dla ZX81

Nazwa programu utworzona została od słów angielskich VERIFY (sprawdzanie) i TAPE LIST (wydruk zawartości taśmy). VTL składa się ze 144 bajtów kodu maszynowego mikroprocesora Z80 i realizuje dwie funkcje:

VERIFY: RAND USR 16514 — weryfikacja poprawności nagrania magnetofonowego przez porównanie kodów odczytywanych z taśmy, z zawartością pamięci mikrokomputera (z pominięciem zmiennych systemowych). Komunikaty: 0/0 — nagranie poprawne; E/0 — błąd (ang. error).

TLIST: RAND USR 16620 — wyświetla na ekranie zawartość taśmy (tytuły kolejnych programów). Każdorazowo po odczytaniu z taśmy tytułu programu przez kilka sekund wyświetlany jest on na ekranie TV, po czym program VTL kontynuuje czytanie taśmy. Pracę programu można w dowolnym momencie przerwać klawiszem **BREAK.** Pojawiające się na ekranie znaki graficzne, a zwłaszcza czarna kreska, świadczą o uszkodzeniu zapisu na taśmie lub niewłaściwym ustawieniu potencjometru głośności magnetofonu.

```
16 FF CD E7 02 CD A3 3A
1B FB C9 0E 01 06 00 3E
7F DB FE D3 FF 1F 30 42
17 17 38 21 10 F1 F1 BA
30 46 CD A3 3A CB 42 79
28 44 17 30 F5 21 09 40
42 CD A3 3A 79 BE 20 24
CD FC 01 18 F3 D5 1E 94
06 1A 1D DB FE 17 CB 7B
7B 38 F5 10 F5 D1 20 04
FE 56 30 B9 3F CB 11 30
B4 C9 CF 0C D5 EB 21 7D
40 37 ED 52 EB D1 30 D0
CF 0D 16 FE 18 94 CB 77
28 02 3E 80 D7 CB 11 30
A9 3E 76 D7 D5 CD 07 02
1E 96 CD 46 0F D2 A6 03
10 F8 1D 20 F5 D1 18 DC
```

Wydruk 1

Program VTL przedstawiono na wydruku 1 w postaci listy bajtów (w kodzie szesnastkowym). Do jego wprowadzenia należy wykorzystać program przedstawiony na wydruku 2. Wpisujemy go z klawiatury i uruchamiamy przyciskiem **RUN.** Możemy teraz wprowadzać kolejne bajty programu VTL z wydruku 1. Jednorazowo można wprowadzać dowolną liczbę bajtów (najwygodniej wierszami), przy czym należy zwrócić uwagę na to, że:

— liczba wprowadzonych przed każdym wciśnięciem klawisza NEWLINE cyfr szesnastkowych musi być parzysta — pomiędzy cyframi nie może występować spacja.

```
 1 LET ABCDEFG=0+0+0+0+0+0+0+0+0+0+0+0+0+0+0+0
 2 POKE 16510,0
 3 POKE 16513,234
 4 LET B=16514
 5 INPUT A$
 6 FOR J=0 TO LEN A$-1 STEP 2
 7 POKE B+J/2,CODE A$(1)*16+CODE A$(2)-476
 8 LET A$=A$(3 TO )
 9 NEXT J
10 LET B=B+J/2
11 IF B<16658 THEN GOTO 5
```

Wydruk 2

Program VTL pamiętany jest w wierszu: 0 REM... . Ponieważ w programie VTL występuje kod NEW LINE (76 HEX=118 DEC), zatem w przypadku automatycznego wydruku wyświetlany jest tylko fragment wiersza 0 REM... Niedogodność tę można usunąć instrukcją:

POKE 16419, 1 [NEW LINE]

Program VTL należy przetestować z magnetofonem. Jeśli nie popełniliśmy błędu przy jego wprowadzaniu, to wiersze 2—11 (wydruk 2) można usunąć, a na ich miejsce wprowadzić program z wydruku 3.

```
 3 PRINT """VTL""  144 BAJTY";TAB 0;"ŁADOWAC OD ADRESU?"
10 INPUT A
11 IF A<8192 THEN LET A=16514
12 FAST
15 LET L=A+11
20 LET H=INT (L/256)
25 LET L=L-256*H
30 FOR I=0 TO 143
35 POKE A+I,PEEK (16514+I)
40 NEXT I
45 POKE A+6,L
50 POKE A+7,H
55 POKE A+35,L
60 POKE A+36,H
65 POKE A+50,L
70 POKE A+51,H
73 PRINT AT 5,0;"USR ";A;" = VERIFY";TAB 0;"USR ";A+106;" = TLIST"
74 SLOW
75 STOP
80 SAVE "VTL"
85 RUN
```

Wydruk 3

Program 3 służy do przemieszczania programu VTL w pamięci. Najwygodniejszymi obszarami do przechowywania tego typu programów są:

— dla komputerów z pamięcią 64 KB — obszar między 8 a 16 K (adresy 8191—16383 DEC)

— dla komputerów z pamięcią od 1 do 16 KB — obszar powyżej zmiennej systemowej RAMTOP.

Przykładowo, dla pamięci 16 KB:

```
POKE 16389, 127    [NEW LINE]
POKE 16388, 112    [NEW LINE]
NEW                [NEW LINE]
LOAD "VTL"         [NEW LINE]
```

Adres, pod którym zostanie załadowany program VTL:

127 * 256 + 112 = 32624

Na pytanie: **LADOWAC OD ADRESU?** należy wprowadzić:

32624 [NEW LINE]

Po załadowaniu programu wyświetlony zostanie komunikat:

USR 32624 =VERIFY
USR 32730 =TLIST

Całość (VTL + program 2) należy zapisać na taśmie instrukcją:

RUN 80 [NEW LINE]

Przed autoweryfikacją nagrania programu VTL, należy wykonać instrukcję **CLEAR**, gdyż po zapisie program zostaje uruchomiony (wiersz 85 RUN), a pamięć mikrokomputera zawiera zmienne i ich wartości, których nie ma na taśmie (zostały skasowane w momencie wydania rozkazu RUN 80.

JAN RABIZO
Agencja Komputerowa „AMICO"
Sosnowiec, tel. 699—649

Mikroprocesor Z80 jest „sercem" wielu popularnych mikrokomputerów. Tym ciekawsze wydaje się doniesienie o dodatkowych rozkazach, rozszerzających powszechnie znany zestaw. Dotyczą one jednak w znacznej mierze operacji wykonywanych na rejestrach IX i IY. Rejestry te mogłyby być bardzo użyteczne, gdyby nie fakt, że ich stosowanie wymaga często 4-bajtowych kodów operacji. Pochłania to pamięć programu i czas procesora. Większość opisanych niżej kodów wymaga tylko dwóch bajtów. Być może przyczyni się to do powszechniejszego stosowania rejestrów IX i IY.

Niepublikowane rozkazy mikroprocesora Z80

Katalogi firm produkujących mikroprocesory Z80 podają 158 grup rozkazowych tego układu. Przeglądając listę rozkazów uporządkowaną według kodów szesnastkowych, można trafić na puste miejsca. Nasuwa się więc pytanie, co zrobi mikroprocesor po napotkaniu jednej z niezdefiniowanych sekwencji bajtów. Okazuje się, że w większości przypadków działanie układu jest sensowne.

Podane dalej informacje sprawdzono na mikroprocesorach firm: NEC, MOSTEK, ZILOG i RFT. Przyjęto następujące dodatkowe oznaczenia:

HX — bardziej znaczący bajt rejestru IX
LX — mniej znaczący bajt rejestru IX
HY — bardziej znaczący bajt rejestru IY
LY — mniej znaczący bajt rejestru IY
F3 — trzeci bit rejestru wskaźników,
F5 — piąty bit rejestru wskaźników.

Rozkaz **SLI** (Shift Left & Increment) ma działanie podobne do SLA, z tym, że na najmniej znaczący bit rejestru podawana jest jedynka. Stan rejestru wskaźników F odpowiada wynikowi operacji, podobnie jak dla rozkazu SLA. Rozkaz **SLI n** ma kod CB3x, gdzie x przyjmuje wartości zależnie od operandu (tabela 1).

Tebela 1

n	A	B	C	D	E	H	L	(HL)
x	7	0	1	2	3	4	5	6

Grupa rozkazów występujących po kodach DD lub FD zawiera najwięcej rozszerzeń w stosunku do listy publikowanej w katalogach. Wszystkie rozkazy (z wyjątkiem EX DE,HL) dotyczące rejestrów H, L oraz pary HL mają swe odpowiedniki dla rejestrów HX, LX, HY, LY. Bity w rejestrze wskaźników przy wykonywaniu tych rozkazów ustawiane są identycznie jak dla standardowych instrukcji. W tabeli 2 podano tylko wersję rozkazów dla rejestru IX. Rozkazy dotyczące IY uzyskamy zastępując DD, w kodzie rozkazu, przez FD.

Rozkazy znalezione w grupie „po DDCB/FDCB" wydają się być ubocznymi efektami działania standardowych instrukcji mikroprocesora, jednakże ich wykorzystanie może być czasem użyteczne. Argumentem rozkazów jest zawsze bajt o adresie IX+d (lub IY+d), natomiast wynik jest przesyłany zarówno do bajtu o adresie IX+d (lub IY+d), jak i do jednego z rejestrów wewnętrznych.

Na przykład — rozkaz RLC(IX+ +01),B (kod DDCB0100) powoduje wykonanie standardowego rozkazu RLC (IX + 01) (kod DDCB010E) oraz przesyłanie wyniku operacji do rejestru B. Analogicznie działają pozostałe rozkazy, tzn. RRC, RLC, RR, RL, SLA, SRA, SLI (również!), SRL oraz SETn i RESn.

Do jednego z rejestrów wewnętrznych jest przesyłana zawsze zawartość wpisywana do bajtu (IX+d) lub (IY+d) po wykonaniu operacji (również w wyniku operacji SET i RES).
Format rozkazów tej grupy jest następujący:

DD CB dd XX lub FD CB dd XX

gdzie XX jest jednym z kodów podanych w tabeli 3.

Rozkazy o bajcie XX równym 01nnnrrr (odpowiadające rozkazom BIT grupy „po CB") działają naturalnie tylko na bitach pamięci zewnętrznej (IX+d) lub (IY+d) — identycznie jak standardowe rozkazy BIT n, (IX+d) lub BIT n, (IY+d).

Napotkanie po bajcie DD lub FD kodu niewymienionego powyżej powoduje zignorowanie bajtu DD lub FD i wykonanie zwykłego rozkazu z listy firmowej.

Na przykład — DDEB powoduje „tylko" EX DE,HL. DD 01 23 45 działa identycznie jak 01 23 45 (LD BC, 4523), a DD ED 7A jak ED 7A (ADC HL,SP).

Bity F5 i F3 w rejestrze wskaźników odzwierciedlają stan odpowiednio trzeciego i piątego bitu wyniku operacji arytmetycznej lub logicznej. Działanie tych bitów sprawdzono na rozkazach operujących na danych ośmiobitowych:

ADD, ADC, SUB, SBC, OR, XOR, AND, CP,

oraz

RLC, RRC, RL, RR, SLA, SRA, SLI, SRL

a także

IN r, (C) — włącznie z IN F, (C).

Przykładowo — niech A = 01110001. Po wykonaniu rozkazu RLC A (kod CB 07) stan akumulatora będzie następujący:

A = 11100010

Bity w rejestrze wskaźników: F5=1, F3 = 0.

Najstarszy, siódmy bit rejestru R nie ulega zmianie przy autoinkrementacji tego rejestru (inkrementowane są tylko bity 0...6), można go więc wykorzystać do pamiętania dowolnych informacji (w wyniku działania rozkazu LD R,A).

PAWEŁ MAĆKÓW
Lubin

Tabela 2

Kod	Proponowana nazwa mnemoniczna	Działanie
DD 24	INC HX	HX←HX+1
DD 25	DEC HX	HX←HX−1
DD 26nn	LD HX, NN	HX←nn
DD 2C	INC LX	LX←LX+1
DD 2D	DEC LX	LX←LX−1
DD 2Enn	LD LX, NN	LX←nn
DD 44	LD B, HX	r_1←r_2
DD 45	LD B, LX	
DD 4C	LD C, HX	
DD 4D	LD C, LX	
DD 54	LD D, HX	
DD 55	LD D, LX	
DD 5C	LD E, HX	
DD 5D	LD E, LX	
DD 6x	LD HX, r	HX←r_2

r	B	C	D	E	HX	LX	A
x	0	1	2	3	4	5	7

DD 6x	LD LX, r	LX←r_2

r	B	C	D	E	HX	LX	A
x	8	9	A	B	C	D	F

DD 7C	LD A, HX	A←HX
DD 7D	LD, A, LX	A←LX
DD 84	ADD A, HX	A←A+HX
DD 85	ADD A, LX	A←A+LX
DD 8C	ADC A, HX	A←A+HX+CF
DD 8D	ADC A, LX	A←A+LX+CF
DD 94	SUB HX	A←A−HX
DD 95	SUB LX	A←A−LX
DD 9C	SBC A, HX	A←A−HX−CF
DD 9D	SBC A, LX	A←A−LX−CF
DD A4	AND HX	A←A∧HX
DD A5	AND LX	A←A∧LX
DD AC	XOR HX	A←A÷HX
DD AD	XOR LX	A←A÷LX
DD B4	OR HX	A←A∨HX
DD B5	OR LX	A←A∨LX
DD BC	CP HX	A−HX
DD BD	CP LX	A−LX

Uwaga: ÷ oznacza różnicę symetryczną (ang. Exclusive OR)

Tabela 3

Mnemonik	Bajt XX	
RLC (IX+d), r	00000rrr	rrr = 000 B
RRC (IX+d), r	00001rrr	001 C
RL (IX+d), r	00010rrr	010 D
RR (IX+d), r	00011rrr	011 E
SLA (IX+d), r	00100rrr	100 H
SRA (IX+d), r	00101rrr	101 L
SLI (IX+d), r	00110rrr	111 A
SRL (IX+d), r	00111rrr	110 (IX+d)
RES n, (IX+d), r	10nnnrrr	
SET n, (IX+d), r	11nnnrrr	

LITERATURA

[1] Weitere U-800-Befehle — Radio, Fernsehen, Elektronik 11, 1983, str. 726,
[2] Unbekannte Z80-Befehle — Elektronik (München), 14, 1980, str. 83
[3] Z80 Assembly Language Programming. L. A. Leventhal — Osborne/McGraw-Hill, 1979.

Przy uruchamianiu programu najwięcej czasu pochłania wyszukiwanie błędów. Najbardziej irytujące jest jednak — poprawianie ich. Nawet gdy program w wersji źródłowej przechowywany jest w pamięci operacyjnej (nie traci się czasu na ładowanie z dysku lub — o zgrozo! — z taśmy papierowej), czas, w którym program jest powtórnie asemblowany dłuży się w nieskończoność. Problem ten wydaje się eliminować system LIC. Dodatkową zaletą systemu jest asemblacja w trybie konwersacyjnym. Pozwala to na bieżąco korygować pomyłki wynikłe przy zapisywaniu programu. Niestety, nie jesteśmy w stanie zamieścić programu; wszystkich zainteresowanych odsyłamy więc do autora systemu.

LIC
wspomaganie projektowania programów

System LIC (ang. label, instruction, comment) jest przeznaczony do konwersacyjnego projektowania programów w języku ASSEMBLER 8080. System LIC umożliwia asemblację, edycję oraz uruchamianie programów. Proces asemblacji jest jednoprzebiegowy, odwracalny; generuje relokowalny kod wynikowy. Językowo zorientowany edytor (ang. symbolic debugger) jest dostępny po asemblacji, w fazie uruchamiania programu. W konwencjonalnych systemach projektowania modyfikacja programu źródłowego wiąże się z koniecznością jego ponownej asemblacji. System LIC umożliwia bezpośrednią modyfikację kodu wynikowego. Edycja programów wynikowych jest szybka (wykonywana w pamięci operacyjnej) i wygodna (DISASEMBLER zapewnia, że kod wynikowy jest widziany przez użytkownika w postaci źródłowej). Językowo zorientowany EDYTOR, ASEMBLER i DISASEMBLER zapewniają szybką integrację sprzętu z oprogramowaniem.

System LIC jest więc szczególnie atrakcyjny dla użytkowników mikrokomputerów nie wyposażonych w pamięci dyskowe. System LIC spełnia wiele funkcji systemu operacyjnego i może być wykorzystany jako program MONITOR.

ASEMBLER w procesie analizy programu źródłowego generuje kod wynikowy, listę etykiet oraz listę komentarzy. Etykiety i komentarze są przechowywane w pamięci operacyjnej i mogą być modyfikowane konwersacyjnie. DISASEMBLER w procesie syntezy kodu wynikowego, symbolicznych etykiet oraz komentarzy generuje program źródłowy. Program źródłowy można asemblować z dowolnego urządzenia wejściowego lub disasemblować na dowolne urządzenie wyjściowe. Podobnie można wprowadzać lub wyprowadzać kod wynikowy, listę etykiet oraz listę komentarzy. Fizyczne urządzenia wejścia-wyjścia są przyporządkowane strumieniom logicznym. Strumieniowi o numerze 0 przypisano klawiaturę i monitor ekranowy. Liczba strumieni wejścia-wyjścia zależy od konfiguracji systemu mikrokomputerowego.

Konwersacyjna asemblacja pozwala na szybką eliminację błędów składniowych. System LIC sygnalizuje miejsce oraz numer błędu. Asembler wprowadza identyfikację (wartością wskaźnika wprowadzania) wierszy źródłowych oraz produktów asemblacji. Identyfikacja obiektów jest wykorzystywana w procesie edycji. Wskaźnik wprowadzania wyznacza miejsce ładowania kodu i jest zawsze określony przed analizą wiersza źródłowego. Wskaźnik wprowadzania stanowi jednocześnie numer wiersza źródłowego, wartość etykiety, adres rozkazu, danej oraz numer komentarza. Adresowi pamięci operacyjnej przypisuje co najwyżej jeden rozkaz (daną), jedną etykietę, jeden komentarz, a w konsekwencji — jeden wiersz źródłowy.

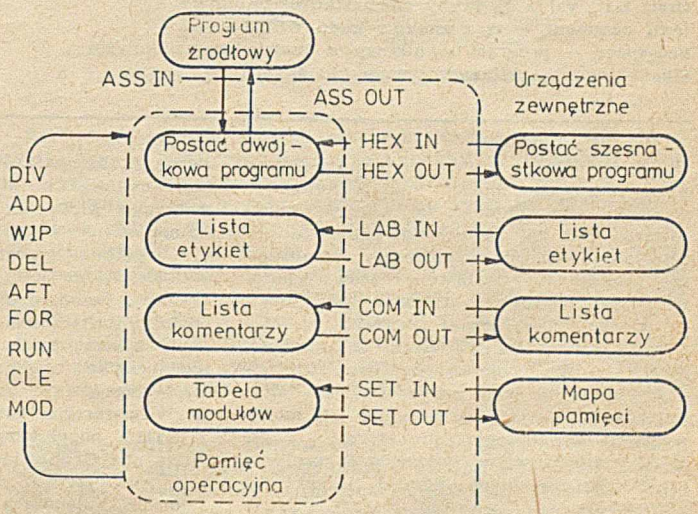
Programy projektowane w systemie LIC składają się ze spójnych obszarów danych i instrukcji. Obszary te nazwano modułami. Modułowa budowa programów umożliwia fragmentaryczną asemblację oraz organizację prostej biblioteki. Moduły stanowiące program można asemblować w dowolnej kolejności. Uruchomione moduły można składować w postaci wynikowej. System LIC dopuszcza składanie programów z modułów źródłowych oraz wynikowych. EDYTOR umożliwia dzielenie, łączenie, usuwanie i relokację modułów wynikowych. Relokacja wiąże się ze zmianą wartości etykiet, zmianą lokalizacji rozkazów (danych) oraz zmianą numeracji komentarzy. W związku ze zmianą wartości etykiet, ulegają zmianie argumenty instrukcji 3-bajtowych, odwołujących się do zmienianych etykiet. Zmiana argumentów rozkazów 3-bajtowych następuje przy przejściu z trybu relokowalnego do trybu absolutnego (MOD). Tryb relokowalny skraca czas edycji oraz umożliwia wykorzystanie niezdefiniowanych jeszcze (ang. forwards) etykiet. Przejście z trybu relokowalnego do absolutnego można traktować jako dopełnienie asemblacji lub konsolidację modułów.

Opis modułu wynikowego składa się z granicy górnej, granicy dolnej obszaru pamięci oraz jednoliterowej nazwy. Nazwy modułów umożliwiają wygodną identyfikację w procesie edycji. Lista opisów modułów (SET OUT) tworzy mapę pamięci operacyjnej. Moduły można wprowadzać (ASS IN) lub definiować (HEX IN, LAB IN, COM IN, SET IN). W pierwszym przypadku produkty asemblacji tworzą treść modułu. W drugim przypadku zawartość modułu jest określana stopniowo poprzez wprowadzanie zawartości pamięci, etykiet, komentarzy i parametrów modułu. Nieznany program binarny można — przykładowo — wprowadzić do pamięci (HEX IN), opisać etykietami (LAB IN), komentarzami (COM IN), a następnie wyprowadzić w postaci źródłowej (ASS OUT). Standardowe etykiety symboliczne są generowane w następstwie dyrektywy SET IN.

Moduły są składowane w postaci źródłowej (ASS OUT) lub w postaci produktów asemblacji (HEX OUT, LAB OUT, COM OUT, SET OUT). Obydwie metody są równoważne.

Program wynikowy jest składowany w kodzie absolutnym (HEX OUT) lub w postaci relokowalnej (HEX OUT, SET OUT). Program wynikowy jest ładowany w miejsce robocze, które zajmował przed składowaniem. Moduły programu relokowalnego można przesunąć na miejsce docelowe (AFT, FOR). Modułom relokowalnym można przypisać symboliczne odwołania wewnętrzne (LAB). Moduły absolutne dopuszczają jedynie symboliczne odwołania wewnętrzne.

System LIC posiada dwa wejścia. Pierwsze służy do inicjowania systemu, natomiast drugie można stosować jako wyjście z programu użytkownika. Drugie wejście oraz procedury monitorowania, dowolnie definiowane przez użytkownika, zastępują w systemie LIC konwencjonalne punkty wstrzymania (ang. break point). Udostępnia ono wszystkie dyrektywy systemu. Wprowadzenie poprawek nie wymaga ponownej asemblacji programu.



Przepływ informacji w systemie LIC

Programy źródłowe oraz wynikowe projektowane w systemie LIC zachowują standard firmy INTEL. Program systemu LIC przeznaczony dla mikrokomputerów wykorzystujących mikroprocesor 8080 zajmuje 6 KB pamięci ROM. Implementacja programu jest uwarunkowana jedynie konfiguracją urządzeń zewnętrznych.

```
LIC-system   := (command)+
command      := / ASS IN  nb first,§
                  (data-space/instr-space)
                / HEX IN  nb
                  (hex-space)
                / LAB IN  nb
                  (label:value)+
                / COM IN  nb
                  (value;comment)+
                / SET IN  nb
                  (§,first,last)+
                / ASS OUT nb §,first,last
                / HEX OUT nb
                  (§,first,last)+
                / LAB OUT nb
                / COM OUT nb
                / SET OUT nb
                / DIV §,value,%
                / ADD §,%
                / WIP §
                / DEL §,first,last
                / AFT value,§
                / FOR value,§
                / RUN value
                / CLE
                / MOD
data-space   := (     declaration
                /     declaration;comment
                / label:declaration;comment )+
instr-space  := (     instruction
                /     instruction;comment
                / label:instruction;comment )+
declaration  := / DB ('text'/ const )
                / DW const
                / DS const
instruction  :=  CMA/CMC/DAA/DI/EI/HLT/NOP/PCHL/
                 RAL/RAR/RC/RET/RM/RNC/RNZ/RP/
                 RPE/RPO/RRC/RZ/SPHL/STC/XCHG/XTHL/
                (ADC/ADD/ANA/CMP/DCR/
                 INR/ORA/SBB/SUB/XRA) (A/B/C/D/E/H/L/M) /
                (INX/DCX/DAD) (B/D/H/SP) /
                (POP/PUSH) (B/D/H/PSW) /
                (LDAX/STAX) (B/D) /
                 MOV (A/B/C/D/E/H/L/M),(A/B/C/D/E/H/L/M) /
                 MVI (A/B/C/D/E/H/L/M),    const /
                (ACI/ADI/ANI/CPI/IN/ORI/
                 OUT/RST/SBI/SUI/XRI)      const /
                (CALL/CC/CM/CNC/CNZ/CP/CPE/CPO/CZ/
                 JCC/JM/JMP/JNC/JNZ/JP/JPE/JPO/JZ/
                 LDA/LHLD/SHLD/STA) label
                 LXI (B/D/H/SP),    label
```

Składnię systemu LIC podano poniżej. Oto objaśnienia zastosowanych zapisów. Metasymbole : = , (,) , + , / oznaczają kolejno: definicję, nawias otwierający, nawias zamykający, rozkaz powtórzenia skończoną liczbę razy, lub.

first, last, value — liczby szesnastkowe 16-bitowe
text, comment — ciąg znaków kodu ASCII
hex-space — program wynikowy w kodzie szesnastkowym
label — symbol alfanumeryczny (do 5 znaków)
const — liczba kodowana dwójkowo, ósemkowo, dziesiętnie lub szesnastkowo
§, % — dowolny znak kodu ASCII
nb — numer strumienia (urządzenia zewnętrznego)

ASS IN — asemblacja tekstu źródłowego. Określenie wskaźnika wprowadzania wartością **first**. Utworzenie modułu wynikowego o nazwie § oraz granicy górnej **first**. Moduł jest wypełniany w miarę wprowadzania kolejnych wierszy źródłowych

ASS OUT — disasemblacja fragmentu **first-last** modułu wynikowego o nazwie §

HEX IN — ładowanie pamięci operacyjnej kodem szesnastkowym

HEX OUT — składowanie zawartości fragmentu **first-last** pamięci operacyjnej w kodzie szesnastkowym

SET IN — definicja modułu wynikowego o nazwie § oraz granicach **first, last**. Obszar **first-last** traktowany jest jako spójny obszar (danych, instrukcji) w zależności od przyjętej nazwy modułu (znak nieliterowy, litera). W przypadku spójnego obszaru instrukcji sporządzana jest lista argumentów instrukcji 3-bajtowych. Generator etykiet przypisuje argumentom standardowe nazwy

SET OUT — składowanie parametrów §, **first, last** wszystkich modułów wynikowych

LAB IN — definicja etykiet o nazwach **label** i wartościach **value**. Jeżeli etykieta o wartości **value** istniała, to przyporządkowana jej zostanie nowa nazwa **label**. Jeżeli etykieta o nazwie **label** istniała, to przyporządkowana jej zostanie nowa wartość **value**

LAB OUT — listowanie etykiet w kolejności alfanumerycznej

COM IN — definicja komentarzy o numerach **value** i treściach **comment**. Poprzednia treść komentarza, o ile istniała, jest gubiona

COM OUT — listowanie komentarzy w kolejności numeracji

DIV — podział modułu § wartością **value** na moduły § i %

ADD — połączenie modułów stycznych §, % w moduł §

WIP — usunięcie modułu §

DEL — usunięcie fragmentu **first-last** modułu §

AFT — przesunięcie modułu § tak, aby jego granica górna pokryła się z wartością **value**. Jeżeli wartość **value** dotyczy obszaru zajętego przez inny moduł %, to moduł § zostanie wstawiony do wnętrza %. Dolna granica modułu % przesunie się, a moduł § przestanie istnieć

FOR — przesunięcie modułu § tak, aby jego dolna granica pokryła się z wartością **value**. Jeżeli wartość **value** dotyczy obszaru zajętego przez inny moduł %, to moduł § zostanie wstawiony do wnętrza %. Górna granica modułu % przesunie się, a moduł § przestanie istnieć

RUN — rozpoczęcie wykonywania programu od adresu **value**

CLE — ustawienie stanu początkowego systemu LIC

MOD — zmiana trybu relokowalny-absolutny. Tryb relokowalny oznaczany jest gwiazdką, a tryb absolutny krzyżykiem. Znaki te wyprowadzane są na monitor przy każdorazowym zgłoszeniu się systemu. Dyrektywy **ASS IN, WIP, DEL, FOR, AFT** dozwolone są tylko w trybie relokowalnym. Dyrektywy **RUN, MEX OUT** dozwolone są tylko w trybie absolutnym.

BOGDAN MICHALAK
Warszawa
tel. sł.: 23-70-81 w. 237
tel. dom.: 39-52-96

● W związku z często zadawanym nam pytaniem, gdzie można nabyć komputer osobisty, odpowiadamy: w każdym domu towarowym... na zachód od Łaby. Zainteresowanym zakupem polecamy opracowanie „Komputery Osobiste" rozpowszechnione przez Branżowy Ośrodek Informacji Naukowej i Technicznej (BOINTE, ul. Marynarska 10, 02-674 Warszawa, tel. 47-07-62). Można w nim znaleźć opisy większości liczących się na rynkach zachodnich komputerów osobistych, porównanie ich możliwości, a nawet tegoroczne ceny.

● Przedsiębiorstwo Polonijno-Zagraniczne „KAREN" zapowiada montaż i sprzedaż mikrokomputera zgodnego z IBM PC. W komplecie przewidziano dwa napędy dysków elastycznych 5.25" oraz drukarkę graficzną (kolorową!). Kontakt: Warszawa, ul. Krochmalna 2 m. 127, tel. 20-11-42.

● Uwaga wielbiciele gier komputerowych! Zamieszczona w INFORMATYCE propozycja wysyłania programów gier w języku FORTRAN nie jest już, niestety, aktualna. W tej sytuacji przekazujemy radę Leszka Drożdża z Rzeszowa polecającego w swym liście książkę Lecha Pijanowskiego „Przewodnik gier", zawierającą opisy wielu gier strategicznych.

● W czwartym mikroKLANIE w rubryce FORTHA popełniono błąd — podając, że bitwa pod Grunwaldem odbyła się we wtorek. Autor tekstu zapomniał o późniejszej reformie kalendarza. Sądzimy jednak, że zwolennicy FORTHA bez trudu zmodyfikują przedstawiony program.

mikroKLAN
nie jest instytucją
klubem
grupą interesu
jest propozycją dialogu

— prowadzi
Andrzej J. Piotrowski
tel. dom.: 48-22-85

W poniższym tekście nie odkrywamy Ameryki. Dla doświadczonych konstruktorów wymyślenie podanych niżej rozwiązań nie przedstawia najmniejszego problemu. Jednak — jak wskazują listy i telefony do redakcji — wśród czytelników mikroKLANU wielu stawia dopiero pierwsze kroki...

Biorąc pod uwagę ustawiczne niedostatki rynkowe, warto przekonać nowicjuszy, że brak jakiegoś układu nie jest jeszcze powodem do załamywania rąk. Wystarczy odrobina inwencji i zawsze znajdzie się jakieś wyjście. Czasem trochę gorsze, innym razem... znacznie lepsze.

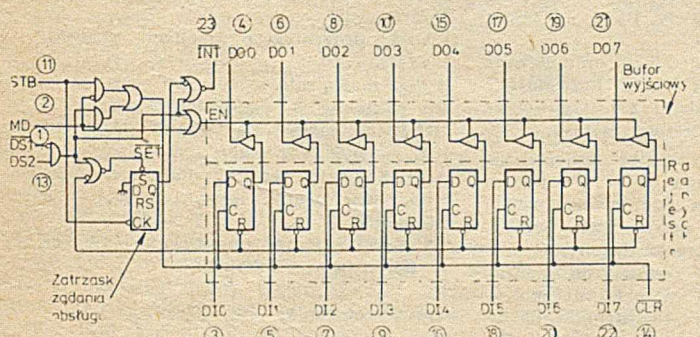
8212 zastępuje 8255 w trybie pierwszym

Układ 8212 wykorzystywany jest najczęściej jako bufor wejściowy lub wyjściowy. Zawarte w jego strukturze zatrzaski pozwalają na zapamiętanie informacji wysłanej przez procesor (wyjście) lub urządzenie zewnętrzne (wejście). Ważną cechą układu jest trzystanowe wyjście (tzw. trzeci stan, nazywany też stanem dużej impedancji, polega na tym, że układ nie wymusza na wyjściu żadnego stanu logicznego — tak jakby nie był w ogóle podłączony), co pozwala na współpracę z magistralą danych systemu mikroprocesorowego.

Konstruktorzy często zbyt pochopnie rezygnują z wykorzystania układu 8212, stosując bardziej „wyrafinowany" układ 8255. Tymczasem w przypadku, gdy w systemie stosowane jest pojedyncze strobowane wejście lub wyjście — z powodzeniem można wykorzystać układ 8212, który jest sporo tańszy od 8255. Przyjęcie takiego rozwiązania ma następujące zalety:

- 8212 wykonany jest w technologii TTL, a więc ma dziesięciokrotnie większą obciążalność wyjść niż 8255
- 8212 nie wymaga programowania i pracuje w żądanej konfiguracji od momentu włączenia zasilania.

Do wad układu należy brak możliwości programowego stwierdzenia źródła przerwania (może to spowodować konieczność zastosowania kontrolera przerwań).

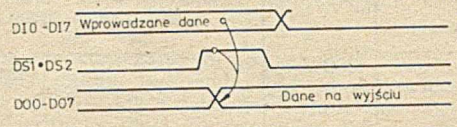


Rys. 1. Struktura logiczna układu 8212

Na rysunku 1 przedstawiono wewnętrzną organizację logiczną układu 8212. Przepływ danych sterowany jest za pomocą wejść: MD (wybór trybu pracy), STB (sygnał strobujący) i DS1/, DS2 (linie wybierające). Układ jest wybierany (bufory wyjściowe nie są w trzecim stanie), gdy na wejście DS1/ podany zostanie poziom niski i na wejście DS2 — poziom wysoki lub gdy na wejście MD podawany jest poziom wysoki. W tym drugim przypadku układ pracuje w sposób ciągły jako wyjście, a dane wpisywane są do zatrzasków sygnałami z linii wybierających. Stan wejścia STB nie ma wtedy wpływu na przepływ danych.

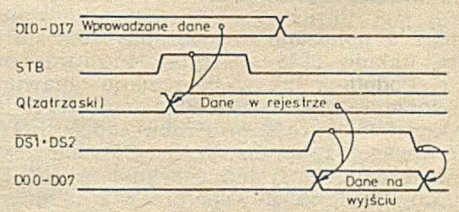
Jeśli na wejście MD podany zostanie poziom niski, dane do zatrzasków wpisywane są sygnałem STB, niezależnie od stanu linii wybierających układ. Linie wyboru sterują jedynie buforami wyjściowymi. Układ pracuje więc w trybie wejściowym.

Zatrzaski mogą zostać wyzerowane w dowolnym momencie przez podanie niskiego poziomu na wejście CLR/.



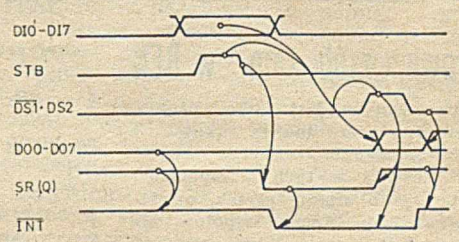
Rys. 2. Przebiegi czasowe dla trybu określonego przez MD=1

Na rysunku 2 podane zostały przebiegi czasowe przy wyborze trybu pracy przez podanie na wejście MD poziomu wysokiego. Na rysunku 3 pokazano natomiast przebiegi czasowe, gdy na MD podany jest poziom niski.



Rys. 3. Przebiegi czasowe dla trybu określonego przez MD=0

Jeżeli na wejściu STB nastąpi zmiana poziomów z wysokiego na niski, to wyjście przerzutnika SR zostanie ustawione w stan 0. Powoduje to wymuszenie niskiego poziomu na wyjściu INT/. Przerzutnik SR ustawiany jest ponownie w stan 1, gdy na wejście CLR/ podany zostanie poziom niski lub gdy na DS1/ podane zostanie 0 i na DS2 — 1. Na rysunku 4 pokazano przebiegi czasowe związane z generacją sygnału INT/.

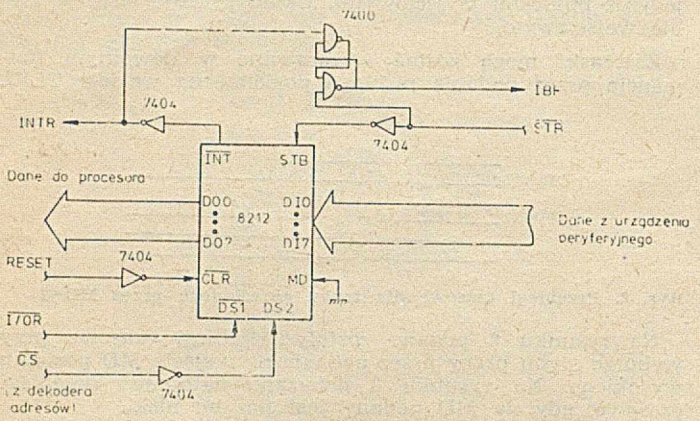


Rys. 4. Zależności czasowe dla sygnału generacji przerwań

Za pomocą kilku dodatkowych bramek można utworzyć konfigurację odpowiadającą współpracy z układem 8255 w trybie 1, gdy został on zaprogramowany jako wejście. Przykładowe rozwiązanie pokazano na rysunku 5. Przebiegi czasowe różnią się tylko dla sygnału zgłoszenia przerwania INTR (aktywny w stanie 1). Dla układu 8255 przechodzi on w stan pasywny 0 — po przejściu strobu odczytu I/OR/ w stan 0 (aktywny). W proponowanej konfiguracji sygnał INTR przechodzi w stan 0 dopiero z chwilą zakończenia strobu odczytu. W praktyce nie ma to większego znaczenia, gdyż zakończenie procedury obsługi przerwania wymaga z reguły wykonania jeszcze kilku instrukcji po odczycie danej.

Brak możliwości bezpośredniego odczytywania statusu praktycznie eliminuje programowy sposób realizacji współpracy (bez wykorzystywania przerwań). Jest to chyba najpoważniejsza wada przedstawionego rozwiązania, gdyż dodanie dodatkowego układu z trzystanowym wyjściem (np. 8216) znacznie podraża koszty konstrukcji. Warto jed-

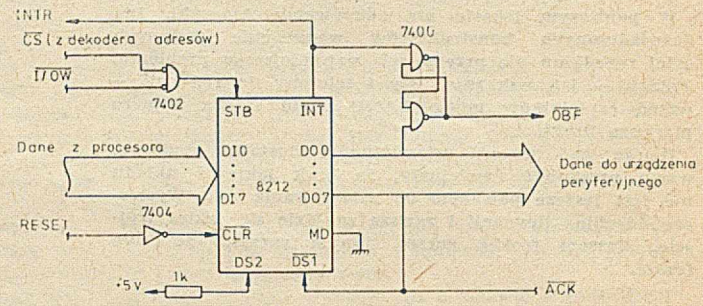
nak dodać, że stosowanie pierwszego trybu pracy dla układu 8255 jest uzasadnione, gdy zakłada się korzystanie z przerwań.



Rys. 5. Układ strobowanego wejścia

Na rysunku 6 pokazano rozwiązanie pozwalające przybliżyć pracę układu 8255 w trybie 1 zaprogramowanego jako wyjście. Podobnie jak w poprzednio omawianym rozwiązaniu, zgłoszenie przerwania sygnałem INTR jest likwidowane dopiero po wycofaniu strobu zapisu I/OW/. Warto też zwrócić uwagę, że przerwanie zgłaszane jest natychmiast po wyzerowaniu systemu sygnałem RESET. Jeżeli nie zamierzamy realizować współpracy z urządzeniem peryferyjnym od samego początku pracy systemu, należy przed wykonaniem przez procesor rozkazu EI ustawić w kontrolerze przerwań odpowiednią maskę. Innym sposobem może być wykorzystanie, faktu, że zgłoszenie przerwania zostanie wycofane po wykonaniu dowolnego zapisu do układu 8212. Metoda ta może być jednak wykorzystana tylko w tych przypadkach, gdy istnieje pewność, że urządzenie peryferyjne nie wprowadzi „fałszywej" danej.

Układ 8255 pracujący w trybie 1 steruje linie wyjściowe niezależnie od stanu podawanego na wejście ACK/.



Rys. 6. Układ strobowanego wyjścia

W przedstawionym rozwiązaniu linie wyjściowe sterowane są tylko w okresie, gdy na linii ACK/ podawany jest poziom niski (aktywny). Odpowiada to sposobowi wyprowadzania danych przez układ 8255 przy pracy w trybie 2. Jeżeli współpraca z urządzeniem peryferyjnym wymaga sterowania linii danych przed podaniem niskiego poziomu na linię ACK/, należy dodatkowo zastosować układ generujący krótki impuls przy przejściu ACK/ z poziomu niskiego na wysoki.

Nie istnieje niestety jednolity protokół przesyłania informacji z obustronnym potwierdzeniem transmisji (ang. handshake). Dlatego też dla różnych urządzeń peryferyjnych może okazać się konieczne dodatkowe zmodyfikowanie przedstawionych propozycji.

AJP

LITERATURA

[1] 8 Bitowa Brama Wejście-Wyjście UCY 74S412. ITE; Aleja Lotników 32/46, 02-663 Warszawa
[2] Kane J., Osborne A.: An Introduction to Microcomputers, vol. 3
[3] The 8080A/9080A MOS Microprocessor Handbook. Advanced Micro Decices, 1977.

Komputerowe „zrób sam" w RFN

W RFN oferowanych jest kilka różnych mikrokomputerów, które można zakupić w częściach, a następnie samemu złożyć i uruchomić. Każdy z tych komputerów pomyślany jest jako zestaw płytek, o różnorodnych funkcjach. Cechą wspólną poniższych propozycji jest ten sam standard magistrali (ECB). Nie wiemy jednak, czy pakiety z różnych zestawów będą ze sobą współpracować.

● **mc — CP/M — Computer**

(ceny w markach; p — płytka, i — instrukcja, k — zestaw do montażu, g — zmontowany i uruchomiony pakiet)

SYS1 zawiera CPU, 64K RAM, 4K EPROM, bootstrap dla systemu operacyjnego CP/M; p+i — 69.—; k — 398.—; g — 498.—; EPROM (zawiera MONITOR) — 39.—

FLO1 sterownik do napędów dysków elastycznych 5.25" (podwójna gęstość) i 8" (pojedyncza gęstość); p+i — 69.—; k — 398.—; g—498.—; dyskietka zawierająca system operacyjny CP/M — 398.—

OUT1 zestaw szeregowych i równoległych wejść—wyjść (2 × V24 i 20 linii TTL); p+i — 65.—; k — 293.—; g — 398.—

TERM1 pakiet sprzęgu graficznego umożliwiający rozdzielczość 256 × 512 punktów, wykorzystujący dwa procesory (Z80 i GDP 9366); i — 20.—; p + 8 K ROM — 149.—; k — 698.—; g — 950.—

W numerze 4/84 pisma „mc" opublikowano sposób modyfikacji FLO1, pozwalający współpracować z napędami 8" przy wykorzystaniu podwójnej gęstości zapisu.

● **NDR — Klein — Computer** jest podstawą kursu prowadzonego przez stację telewizyjną NDR (oprócz podręcznika-instrukcji montażu można nabyć dwie wideo-kasety z nagranym kursem telewizyjnym — pierwsza cena dotyczy zestawu do montażu, druga uruchomionego pakietu; można też nabywać osobno płytki drukowane w cenie 15 marek)

SBC2 Z80A CPU, RAM, gniazdo pod EPROM — 79.95/129.— [M, 1]

IOE równoległe we-wy 2 × 8 bitów — 39.95/69.— [1]

KEY sprzęg równoległy do klawiatury — 49.95/89.— [1]

GDP 64K sprzęg graficzny z wykorzystaniem GDP 9366 — 269.—/378.— [1]

CAS sprzęg z magnetofonem kasetowym — 74.90/129.— [1]

POW5V zasilacz +5V/3A — 39.95/58.50.— [M, 1]

CPU68K pakiet procesora 68008 — 199.—/265.— [2]

ROA64 pakiet pamięci 8K × 8 RAM/ROM — 39.95/84.— [2]

PROM programator pamięci EPROM — 79.95/129.— [2]

BUS2I magistrala systemowa ECB dla sześciu pakietów — 39.95/58.50 [1]

MINIBUS magistrala dla POW5V, SBC2, IOE — 19.50/24.50 [M]

IOE-EX pakiet we—wy dla sterowania robota, generowania muzyki i in. — 89.90/145.— [M]

TAST klawiatura (montowana fabrycznie) — 198.— [1]

Oprogramowanie:

MON1 MONITOR dla SBC2 — 44.50 [1]
BASIC dla SBC2 — 50.—
MON68K MONITOR/ASSEMBLER dla CPU68K — 155.— [2]
PASCAL dla CPU68K — 155.—

Oferowane są również zestawy:

PAK-M zestaw eksperymentalny z pakietów oznaczonych [M] — 219.—/333.—
PAK-1 pakiety i oprogramowanie oznaczone [1] — 799.—/1095.—
PAK-2 pakiety i oprogramowanie oznaczone [2] — 555.—/699.—
PAK-3 pakiety i oprogramowanie oznaczone [1] lub [2] — 1349.—/1794.—